UNIVERSAL MULTIHEAD AUTOMATA

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

By

Daniel Paul Martin

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

in the School of Information and Computer Science

Georgia Institute of Technology

June, 1974

UNIVERSAL MULTIHEAD AUTOMATA

Approved:

John M. Gwynn, Jr., Chairman

Lucio Chiaraviglio

William J. Grosky

Richard Y. Kain

Date approved by Chairman: 5/23/74

For Shahrnaz

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Professor John M. Gwynn, Jr., for his intellectual stimulation, guidance, encouragement and uncanny ability to ask interesting, thought-provoking questions.

I would also like to thank the members of my doctoral guidance committee, Professors Chiaraviglio and Grosky, for their invaluable advice.

Professor Richard Y. Kain of the University of Minnesota is especially thanked for his encouragement upon reading a preliminary portion of these results and for his help as an outside reader of this thesis.

My graduate study was supported in part by a teaching assistantship in the School of Mathematics at the Georgia Institute of Technology. Special thanks are due to the late Professor George C. Caldwell for helping make this aid available.

TABLE OF CONTENTS

GLOSSARY

ϵ	(i) set membership; (ii) null string.
A^*	Kleene closure of the set A.
$\subset$	(i) proper set inclusion; (ii) induced relation on automata based upon the languages which they accept.
$\underline{\subset}$	(i) set inclusion; (ii) same as (ii) above.
=	(i) ordinary equality; (ii) equivalence when comparing classes of automata.
T(M)	acceptance set of the machine M.
FA	finite automata.
PDA	pushdown automata.
CA	counter automata.
LBA	linear bounded automata.
aux-PDM	auxiliary pushdown machine.
aux-PDM,L(n)	auxiliary pushdown machine operating in space L(n).
TM	turing machine.
TM,T(n)	turing machine operating in time T(n).
TM,L(n)	turing machine operating in space L(n).
Σ_M	input alphabet of machine M.
Γ_M	auxiliary memory alphabet of machine M.
RL	regular language.
CFL	context-free language.
CSL	context-sensitive language.
REL	recursively enumerable language.

α_M	description of machine M.
E_M	input encoding of machine M.

CHAPTER I

INTRODUCTION

Definitions and Notation

Automata theorists have recently devoted a great deal of effort to the study of multihead automata. A multihead automaton, M, consists of (i) a control which is in one of a finite number of states; (ii) an input tape containing a finite string over a finite alphabet, Σ_M; (iii) a finite number of read heads scanning symbols on the input tape; (iv) an auxiliary memory containing a finite number of symbols from a finite alphabet, Γ_M; and (v) a means of access to the auxiliary memory.

The machine makes "moves" based upon a portion of its current configuration, namely the state, the contents of a finite portion of the auxiliary memory and the h-tuple of input symbols under the h heads. A move reflects (i) a possible state transition, (ii) a modification of some finite portion of the auxiliary memory and (iii) independent motion of each read head. If the machine is one-way, then each read head can move one square right or remain stationary. Each head of a two-way machine can also move one square left.

Machines are classified according to the size and accessibility of the auxiliary memory. This memory is viewed as some fixed number of storage tapes upon which the automaton can write symbols as well as read them; there is a single read/write head on each tape. A

Turing machine, TM, has an unrestricted auxiliary memory; that is, any symbol on any tape can be overwritten and there is no prior bound on the potential number of squares that can be scanned on any tape. If any scanned symbol can be overwritten and, for inputs of length n, only L(n) distinct symbols can be used on any tape, then the associated machine is said to be a TM operating in space L(n), TM, L(n). The case in which L(n) is a linear function has received special consideration; such a machine is called a linear bounded automaton, LBA.

The pushdown automaton, PDA, and the stack automaton, SA, both have a single storage tape infinite to the right. Initially, the leftmost square contains a special symbol, called the bottom, and the rest of the tape is blank. Access to this tape is limited, however.

In a PDA this tape is called the "store," and the read/write head is always located at the rightmost non-blank symbol on the tape. In a single move, the PDA can (i) leave the scanned symbol alone, (ii) erase the scanned symbol and move the read/write head one square left or (iii) move the read/write head one square right and add a symbol to the top of the store.

In a SA this tape is called the "stack," and the read/write head is allowed to move to the left of the rightmost non-blank symbol for reading. Writing or erasure, however, must take place exactly as in a PDA.

A machine which has some finite number of unrestricted storage tapes in addition to a pushdown store is called an auxiliary pushdown machine, aux-PDM [5]. Since the unrestricted version of this machine is equivalent to a TM, a bound is usually placed on the number of distinct

squares that can be scanned on the unrestricted storage tapes, called work tapes; no bound is placed on the potential size of the pushdown store. If, for inputs of length n, no more than L(n) distinct squares can be scanned on any work tape, then the machine is called an aux-PDM, L(n). Mager [19] separately defined a writing pushdown automaton, WPDA, which is an aux-PDM operating in linear space, or equivalently an LBA with a pushdown store. The WPDA has only one TM work tape.

A pushdown automaton which has only one symbol which can be used to overwrite blanks in the store is called a counter automaton, CA [16], and its store is called a counter. A finite automaton, FA, has no auxiliary memory.

If the control of M allows at most one move from any configuration, then M is a deterministic machine. A non-deterministic machine may have more than one allowable move from each configuration.

Machines examined in this thesis can be viewed as language acceptors: when a string x is submitted to a machine M, M either accepts or rejects x in the following sense. The string ¢x$ is placed on the input tape of M; the left endmarker ¢ and the right endmarker $ are symbols in Σ_M which do not appear in x, i.e. $x \in (\Sigma_M - \{¢,\$\})^*$. M will start in a distinguished start state, with some unique initial contents in the auxiliary memory and all of its read heads scanning the ¢. M has a distinguished subset of the set of states, called the set of final states, and if x can ever cause M to enter one of these states when all read heads are scanning the $, then M halts and accepts x. M will also halt if any read head moves to the left of ¢ or to the right of $, if

the auxiliary memory becomes empty or if there is no allowable move from the current configuration. For each machine M, we define $T(M) = \{x|x \text{ is accepted by } M\}$. Machines M and M' are called equivalent if $T(M) = T(M')$.

Machine classes will be designated by names according to the following convention: (i) the first character of the name (1 or 2) indicates whether the machine is one-way or two-way; (ii) the second character (D or N) indicates whether the machine is deterministic or non-deterministic; (iii) the next set of characters (e.g. PDA,SA, FA, ...) classifies the memory of the machine; (iv) the last group of characters ((h), h a positive integer) indicates that the machine has h read heads. If h = 1, this group of characters will be omitted. Moreover, since LBA and TM are always two-way, the initial number is omitted in these cases. Thus, the name 2DPDA(3) stands for the class of two-way deterministic pushdown automata with three read heads, while 1NFA is the class of one-way non-deterministic finite automata with one read head.

Automata have been hierarchically arranged according to the sets they accept. For example, any set which can be accepted by a member of the class 1NFA can also be accepted by some machine in 1DPDA [14]; we say that 1NFA $\subseteq$ 1DPDA. But since the set $\{a^i b^i : i \geq 1\}$ can be accepted by some $M \in$ 1DPDA but by no $M' \in$ 1NFA [7], 1NFA $\subset$ 1DPDA.

Historical Overview

Chomsky [4] gave such hierarchies their initial impetus when he exhibited generators of languages called formal grammars. Chomsky classified these grammars and developed the four traditional language classes: regular languages, RL; context-free languages, CFL;

context-sensitive languages, CSL; and recursively enumerable languages, REL. It was shown that for each class of languages there was an equivalent class of automata; the machines in the class accepted all and only members of the corresponding language class. Corresponding to RL are 1NFA [14], to CFL are 1NPDA [7], to CSL are NLBA [18] and to REL are NTM [14]. It is known that RL $\subset$ CFL $\subset$ CSL $\subset$ REL, or equivalently 1NFA $\subset$ 1NPDA $\subset$ NLBA $\subset$ NTM.

Although formal grammars have not been exhibited for most classes of automata which have arisen since Chomsky's initial work, efforts to establish a hierarchy of automata comprise much of the existing literature and give rise to many interesting (and difficult) unsolved problems in the area. As each new class of automata is defined, questions of inclusion, more specifically proper inclusion and incomparability, immediately arise in trying to put the new class in its proper place among the existing classes.

The classes 2DFA and 2NFA were first examined by Rabin [22]. Sheperdson [26] has shown that the two-way motion of the read head did not increase the acceptance power of the machine, i.e. 2NFA = 2DFA = 1NFA = 1DFA.

The classes 1DFA(h) and 1NFA(h) were introduced by Rosenberg [24]. Rosenberg's definition differs slightly from ours, in that only one head at a time can scan the input tape, but it is easy to show that the two definitions are equivalent. Rosenberg established an infinite hierarchy 1DFA(h) $\subset$ 1DFA(h+1) for h $\geq$ 1. Rosenberg also showed that the context-free language $L = \{x00x^R | x \in \{1,2\}^*\}$ is not in T(M) for any

$M \in \bigcup_{h=1}^{\infty} 1DFA(h)$. Since the context-sensitive language $L_1 = \{a^i b^i c^i | i \geq 1\}$ is accepted by some member of 1DFA(2), it follows that one-way multi-head automata are not comparable to 1NPDA. Rosenberg also showed for each $h \geq 1$ that the languages accepted by 1NFA(h) were not closed under union, intersection or concatenation.

The classes 2DFA(h) and 2NFA(h) have been examined by Ibarra and Chobbam. Ibarra [13] has shown that $2DFA(h) \subset 2DFA(h+2)$ for $h \geq 1$; whether a similar result holds for the class 2NFA(h) remains open. In an unpublished but often referenced [5,13] paper, Chobbam has shown that $\bigcup_{h=1}^{\infty} 2DFA(h) = DTM, L(n) = \log(n)$ and $\bigcup_{h=1}^{\infty} 2NFA(h) = NTM, L(n) = \log(n)$.

The 2DPDA was initially defined by Gray, Harrison and Ibarra [8]. Since it was known that 1NPDA correspond to CFL and that $1DPDA \subset 1NPDA$, a question which naturally arose was whether the two-way motion of the input head compensated for the lack of non-determinacy in the finite control. This question remains open. It is easy to show that there are CSL which can be accepted by members of 2DPDA (e.g. $L = \{a^i b^i c^i | 1 \geq 1\}$ [8]), and it is conjectured that there exists a CFL which cannot be accepted by any $M \in$ 2DPDA [1,14]. If the conjecture could be proved, then the classes 2DPDA and 1NPDA would not be comparable. Gray, Harrison and Ibarra did show, however, that $2DPDA \subseteq DLBA \subseteq CSL$.

The non-deterministic version of this machine, 2NPDA, is clearly more powerful than 1NPDA; thus the sets 2NPDA acceptable properly include the CFL. It is not known whether $2DPDA \subset 2NPDA$. Moreover, results relating 2NPDA to NLBA have not as yet appeared, but it is obvious that $2NPDA \subseteq 2NSA$.

The multihead pushdown automaton was introduced by Harrison and Ibarra [10]. The original paper was primarily concerned with the establishment of closure properties for sets accepted by 2DPDA(h) and 2NPDA(h) and with the relationship of these machines to pushdown automata with several input tapes.

Ibarra [13] established two infinite hierarchies, showing that for $h \geq 1$, 2DPDA(h) $\subset$ 2DPDA(h+1) and 2NPDA(h) $\subset$ 2NPDA(h+1). These hierarchies follow from existence proofs and he does not actually exhibit the required languages.

A relationship between 2NPDA(h) and 2DPDA(h') was established by Kameda [16]. Kameda defined a bounded counter machine with several counters. This machine has a fixed number of counters and no counter can ever contain more non-blank symbols than specified by the bound.

Relationships between the bounded counter machines, multihead PDA and TM allow Kameda to show that 2NPDA(h) $\subseteq$ 2DPDA(12h+1) for each $h \geq 1$. An existence proof is again used; the actual construction of the required 2DPDA(12h+1) is not shown.

Another technique of establishing hierarchies has been used by Cook [5] and involves the use of TM time and space classes. A TM operating in time T(n), TM,T(n), must make no more than T(n) moves before accepting an input of length n. A refinement theorem by Hennie and Stearns [11] can be used to show that the class C is more powerful than the class C' if C' can be shown equivalent to TM,$T_2(n)$, C is equivalent to TM,$T_1(n)$ and $\liminf \frac{T_2(n)\log T_2(n)}{T_1(n)} = 0$.

Cook's work allows the study of several classes of automata in

a unified setting of aux-PDM. He exhibited several classes of automata that were equivalent to space classes of aux-PDM and time classes of DTM. The Hennie and Stearns theorem can then be used to establish the following:

$$\bigcup_{h=1}^{\infty} 2DPDA(h) = \bigcup_{h=1}^{\infty} 2NPDA(h) \subset 2DWPDA = 2NWPDA \subset 2DSA \subset 2NSA$$

Unfortunately, Cook's results that $\bigcup_{h=1}^{\infty} 2DPDA(h) \equiv \bigcup_{h=1}^{\infty} 2NPDA(h)$ can not be used to relate 2DPDA(h) to 2NPDA(h) for a fixed integer h, since the constructions depend upon Chobbam's unpublished result which is dependent on the size of the input alphabets rather than the number of heads.

Cook, Ibarra and Kameda achieved their results with indirect methods. A popular direct method of establishing hierarchies is through the construction of a diagonal language. This method was first used by Knuth and Bigelow [17] to prove that NLBA $\subset$ 2NSA. A diagonal construction of a language not in a particular class C proceeds in the following manner:

i. Let L_1 be a language of descriptions of acceptors for the class C, such that every acceptor in C has at least one description in L_1.

ii. The diagonal language is $L = \{E(x) \mid x \in L_1$ and the machine M described by x does not accept $E(x)\}$. E(x) is a fixed encoding which maps symbol i of the alphabet of L_1 into symbol i of Σ_M.

iii. L cannot be in C because the existence in C of an acceptor

for L would give rise to an obvious contradiction.

A class of machines C' will be more powerful than a class C if there is a machine M' ϵ C' which can accept the diagonal language for C and furthermore C $\subseteq$ C'. If submitted α_M, the description of M ϵ C, M' ϵ C' can obviously operate by simulating M acting on input $\cent\alpha_M\$$, rejecting α_M if M would accept it and accepting α_M if M would reject it. Mager [19] used this method to show that 2DPDA $\subseteq$ 2WPDA; Martin and Gwynn [20] used a similar method to construct a non-regular, non-context-free language $\bar{L}$ which is accepted by a member of 2DPDA.

One might suspect that the non-regular language $\bar{L}$ might fall within the next class in the Chomsky hierarchy; that $\bar{L}$ is not context-free, however, illustrates a problem which often arises in diagonal language construction. Intermediate classes frequently exist between the classes C and C', where the class C' contains the machine accepting the diagonal language of C. Ibarra [13] has also noted such "gaps," particularly when he constructed a 2DFA(2h+3) which was able to accept the diagonal language for 2DFA(h). An additional h + 1 heads were used to determine whether the machine described would enter an infinite loop when submitted its description. Indirect methods were required to achieve his final refinement, 2DFA(h) $\subset$ 2DFA(h+2).

Given a description α_M of a machine M in C, diagonal acceptors can use α_M to simulate the action of M on input $\cent\alpha_M\$$. The diagonal acceptor must also determine, in a finite number of moves, whether the simulation would produce an infinite loop. Such a determination seems to make the construction of a diagonal acceptor more difficult than the construction

of a simulator for the machines in C; for simulation alone, the recognition of an infinite loop is unnecessary.

The diagonal acceptor of C only simulates the behavior of each machine M in C for the input $\phi\alpha_M\$$. A general simulator S must simulate the behavior of each machine for all of its allowable inputs. To effect the simulation of M for input $\phi x\$$, S will act on input $\phi\alpha_M E(x)\$$. The encoding of $x,E(x)$, is used if C contains machines with arbitrarily large alphabets; such is the case for all automata classes previously mentioned.

A simulator of machines in C can easily be made into a machine U which is universal over C: for each M in C and each allowable input x of M, $\alpha_M E(x) \in T(U)$ if and only if $x \in T(M)$. However, a universal machine is not required to perform simulations in order to determine if $\alpha_M E(x) \in T(U)$; U's ability to determine this makes it universal regardless of how the determination is made.

For any class C of machines an interesting question arises. What is the smallest class C' which contains a machine universal over C? It is well known that C' = C for TM [12], and Rahimi [23] has shown C' $\neq$ C for 1NFA, 1DPDA and 1NPDA. The question remains open for C = DLBA and C = NLBA. However, Rahimi [23] and Owings and Feldman [21] have exhibited universal machines for subclasses of DLBA and NLBA with restricted alphabet sizes; these universal machines, although in DLBA or NLBA, fall outside the classes over which they are universal. Martin and Gwynn [20] have constructed a machine in 2DPDA universal over 1NFA; this machine is a simple modification of the corresponding diagonal language

acceptor, and most known diagonal acceptors can be similarly modified to produce universal machines.

Summary of Results

In Chapter II we construct a number of universal machines from the classes 2DPDA(h), 2NPDA(h), 2DFA(h), 2NFA(h), and 2WPDA. It is of interest that all but one of these universal machines requires an input encoding which depends on the machine being simulated.

In Theorem 1 we construct a machine U, a 2DPDA, which is universal over 1NFA. This currently provides the best answer to the question of what is the smallest class C which contains a machine universal over 1NFA. Rahimi [23] expressed interest in this question after he proved that 1NPDA (and consequently, 1NFA) could not be C; any such class C would contain a member which accepts a non-context-free language. Rahimi was perhaps interested in classes C such that 1NPDA $\subset$ C; if, in addition, C $\subset$ NLBA than 1NPDA $\subset$ C $\subset$ NLBA. C would then be a class of acceptors whose corresponding languages lie properly between CFL and CSL. Unfortunately, although 2DPDA $\subseteq$ DLBA $\subseteq$ NLBA, it is still open whether 1NPDA $\subseteq$ 2DPDA or 2DPDA $\subset$ NLBA.

If U were universal over 1NPDA, this fact would not alone imply that 2DPDA accepts a language accepted by no member of 1NPDA; however, Rahimi's result, coupled with the fact that U is only universal over 1NFA, a proper subclass of 1NPDA, is sufficient to show that T(U) is such a language. This situation is somewhat unique among automata hierarchies and seems worthy of note.

We next construct a diagonal language L_4 for 1NFA, and in Theorem

2 show that a slight modification of U will produce a machine D ϵ 2DPDA such that T(D) = L_4. Since any member of 1NFA accepts or rejects a string in no more than n moves, it is not surprising that U forms the basis of a diagonal acceptor of 1NFA.

In Theorem 3 we show that L_4 is not context-free. It is of note that Rahimi's result is not useful in showing that the non-regular language L_4 is not context-free. This diagonal language falls outside the next Chomsky class. Knuth and Bigelow's diagonal language for NLBA fell properly within the next Chomsky class, REL; their non-context-sensitive language was accepted by a member 2NSA $\subseteq$ TM.

We show, in Theorems 4 and 5, that for each h $\geq$ 1, each of the classes 2DPDA(h) and 2NPDA(h) contains a universal member. These results, while definitive concerning universal machine membership, shed no light on diagonal language acceptors. Since no class can contain a member which accepts its own diagonal language, the universal machines constructed clearly cannot be modified to accept diagonal languages for their own classes.

In Corollary 4.1 we show that for each h $\geq$ 1, 2DPDA(h) contains a member universal over 2DFA(h); Corollary 5.1 provides an analogous result for 2NPDA(h) and 2NFA(h). It is still not apparent how to modify these universal machines to accept a diagonal language for 2DFA(h) or 2NFA(h); thus, these results fall short of settling whether 2DFA(h) $\subset$ 2DPDA(h) or 2NFA(h) $\subset$ 2NPDA(h). However, the existence of these machines makes 2DPDA(h) a candidate for the smallest class containing a member universal over 2DFA(h) and, correspondingly, 2NPDA(h) over 2NFA(h).

In Theorem 6 we show, for each $h \geq 2$, that 2DPDA(h-1) contains a member universal over 1DFA(h) and 2NPDA(h-1) contains a member universal over 1NFA(h). These results are based on the fact that for any member of 1DFA(h) there is an equivalent member of 2DPDA(h-1), and correspondingly, for any member of 1NFA(h) an equivalent member of 2NPDA(h-1). These machines with h-1 heads use h-2 one-way heads and one two-way head. It is still not known whether either class of such machines is less powerful than the corresponding unrestricted class with h-1 two-way heads; with this question still unsettled, 2DPDA(h-1) becomes a candidate for the smallest class containing a member universal over 1DFA(h), and, correspondingly, 2NPDA(h-1) for 1NFA(h).

Theorem 6 can now be used to prove Theorem 7: for $h \geq 2$, 1DFA(h) $\subset$ 2DPDA(h) and 1NFA(h) $\subset$ 2NPDA(h). These results follow from Lemma 8, which states that 2DPDA(h-1) cannot contain a member universal over 2DPDA(h), and correspondingly, 2NPDA(h-1) not over 2NPDA(h). Here the universal machines also prove useful in structuring automata classes.

Ibarra [13] has exhibited a member M of 2DFA(h+2), M universal over 2DFA(h); although he did not explicitly do so, his construction could be used to construct a member M' of 2NFA(h+2), M' universal over 2NFA(h). In Theorems 8 and 9 we improve upon Ibarra's result by exhibiting members of 2DFA(h+1) and 2NFA(h+1) universal over 2DFA(h) and 2NFA(h), respectively. Again, however, these results provide no clue as to whether the h-head machines are less powerful than the corresponding h + 1 head machines.

Theorem 8 also makes 2DFA(h+1) a candidate for the smallest class containing a member universal over 2DFA(h); Corollary 4.1 made 2DPDA(h) such a candidate. It is still open whether 2DFA(h+1) and 2DPDA(h) are comparable. Results from Theorem 9 and Corollary 5.1 make 2NFA(h+1) and 2NPDA(h) candidates for the smallest class with respect to 2NFA(h); the question of incomparability is also open for these classes.

Cook [5] showed that $\bigcup_{h=1}^{\infty} 2DPDA(h) = \bigcup_{h=1}^{\infty} 2NPDA(h) \subset 2WPDA$; in Theorem 10 we exhibit a deterministic $U \in 2WPDA$, U universal over 2DPDA(h) for all h. U is clearly universal over 2NPDA(h) for all h, and, moreover, Lemma 8 shows that T(U) cannot be accepted by any member of 2NPDA(h) for any h. Cook showed $\bigcup_{h=1}^{\infty} 2DPDA(h) \subset 2WPDA$ by indirect methods; we have exhibited directly, through universal machines, a language which Cook proved must exist. Such a language had not previously been exhibited directly. Although Mager [19] has shown that for any member of 2WPDA there exists an equivalent member which halts for all input, such a member equivalent to U will not trivially form the basis of a diagonal language acceptor for $\bigcup_{h=1}^{\infty} 2DPDA(h)$. The variable input encoding causes difficulties in obtaining a contradiction when a diagonal construction is attempted.

A trivial refinement of each of the above results is possible. For each universal PDA constructed, we could construct another one whose store alphabet consists of only three symbols and whose input alphabet consists of only four symbols. For each universal FA constructed, we could construct another one whose input alphabet consists of only four symbols.

In Chapter III we consider certain restricted subclasses of the CFL and show that members of 2DPDA(h) accept these for $h \leq 3$. Kameda [16] has shown that 2NPDA $\subseteq$ 2DPDA(13). Since 1NPDA accepts the CFL and 1NPDA $\subset$ 2NPDA, 2DPDA(13) accepts, among other languages, all the CFL. It is, however, possible to accept each minimal linear languages [9] by a member of 2DPDA(2) (Theorem 11), each linear language [2] by a member of 2DPDA(2) (Theorem 12), each k-linear language [25] by a member of 2DPDA(3) (Theorem 13), and each metalinear language [25] by a member of 2DPDA(3) (Corollary 13.1). Some of the CFL which have been conjectured not to be accepted by any member of 2DPDA fall within the subclasses listed above. Although it remains open whether these languages can be accepted by a member of 2DPDA, the above results show that the addition of one or two extra heads is sufficient.

It is known that minimal linear $\subset$ linear(1-linear) $\subset$ 2-linear $\subset$ $\cdots$ k-linear $\subset$ metalinear [25]. Universal machines cannot be directly used to provide a single member of 2DPDA(3) which accepts, if provided a description of the language, any properly encoded metalinear language. However, techniques used in Theorems 11, 12, and 13 might provide such an acceptor in 2DPDA(k) for $k = 4$ or 5 if not for $k = 3$. We have not as yet attempted such a construction but hope to do so in the near future.

CHAPTER II

UNIVERSAL MACHINES

Formal Definitions

A machine U will be universal over a class of machines C if for each M in C there exists $\alpha_M \in \Sigma_U^*$ and an encoding function E_M such that if $x \in \Sigma_M^*$ then U accepts $\alpha_M E_M(x)$ if and only if M accepts x. α_M is usually a description of M or of some M' in C such that T(M) = T(M'). A function $E: \Sigma_M^* \to \Sigma_U^*$ is an encoding function provided there exists a 1-1 function $g: \Sigma_M \to (\Sigma_U - \{|\})^*$ such that for each $x \in \Sigma_M$ and $w \in \Sigma_M^*$, $E(xw) = g(x)|E(w)$ and $E(x) = g(x)$. Thus $E(s_1 s_2 \ldots s_k) = g(s_1)|g(s_2)|\cdots|g(s_k)$ for $s_i \in \Sigma_M$. The | is inserted to separate encoded symbols.

Since the encoding functions for different machines may be different, unrestricted encoding would allow $E_M(x) = 1$ if M accepts x and $E_M(x) = 0$ if M rejects x. Our definition requires that each symbol be encoded individually, independent of its context, and that the order of symbols in a string be preserved. These restrictions prevent $E_M(x)$ from containing any additional information about M's acceptance or rejection of x.

A 2DPDA Universal Over 1NFA

We first define the language L_1 of 1NFA descriptions. A string x over $\{q,s,:,(,)\}$ is in L_1 if and only if it is a finite concatenation of substrings of the form $q^i s^j : q^k)$ or (q^i) for i,j,k positive integers. A substring $q^i s^j : q^k)$ represents an allowed transition from state i to

state k when symbol j is under the read head, while (q^i) indicates that i is a final state.

The endmarkers ¢ and $ are known to be unnecessary for the classes 1NFA and 1DFA; such automata will be considered without endmarkers for the remainder of this thesis.

LEMMA 1. There exists a 2DPDA, M_1, so that $T(M_1) = L_1$.

Proof. In fact, L_1 is a regular language. Consider the regular grammar G whose language is L_1 and whose productions are:

S→(A	S→qC
A→qB	C→qC
B→qB	C→sD
B→)S	D→sD
B→)	E→qB
	D→:E

Q.E.D.

If no transition is allowed from state i when symbol j is being read, substrings of the form $q^i s^j : q^k)$ may not appear in a description $x \in L_1$. Such a description is not complete; we define a language L_2 of complete descriptions. A description $x \in L_2$ if and only if there is a substring, $q^i s^j : q^k)$, in x for each $1 \leq i \leq n(q)$ and $1 \leq j \leq n(s)$ where n(q) and n(s) are the maximum number of consecutive q's and s's, respectively, which appear in x.

LEMMA 2. There exists a 2DPDA, M_2, so that $T(M_2) = L_2$.

Proof. We describe the actions of M_2 as follows.

1. Determine if $x \in L_1$; if not, reject x.

2. Search x for n(s) and place n(s) S's on the store.

Erase S from the store for each s in the current substring of s's

in x. If # is on the store with s under the read head, or if : is under the read head with S on the store, reverse direction, scan to the first s of the current substring of s's, reverse again, and add an S to the store for every s thus encountered until s is no longer under the read head. Continue to the next substring of s's in x. The store now contains $\#S^{n(s)}$.

3. Search x for n(q) and place n(q) Q's on the store. The procedure is similar to step 2, except that S will be encountered on the store instead of #. The store now contains $\#S^{n(s)}Q^{n(q)}$.

4. With S^jQ^i on the store, search x for a substring of the form q^is^j. The search is again performed by erasing corresponding store symbols for input symbols read. Store reconstruction is performed as in step 2 for each non-matching substring. A match causes store reconstruction, erasure of the top Q and a continuing search in the event that Q's remain on the store. If there are no Q's, then the top S is erased: if any S's remain, n(q) Q's are added to the store and search resumes; if no S's remain, x is a complete description and is accepted by M_2. Q.E.D.

It is obvious that the subclass of 1NFA spanned by L_2 spans the class 1NFA. Since 1NFA = 1DFA, it is trivial to show that 1DFA with complete descriptions span 1NFA. We therefore define the language L_3 of complete and minimal descriptions of 1DFA. If $x \in L_3$, then for each $1 \leq i \leq n(q)$ and $1 \leq j \leq n(s)$, there is exactly one substring q^is^j: in x, moreover, if $1 \leq i \leq n(q)$ then the substring (q^i) may not appear more than once.

We have required that L_3 contain only descriptions of DFA to make the construction of U easier; the requirement that no superfluous element appear in the description is later used to show that a certain subset of L_3 is not context-free.

LEMMA 3. There exists a 2DPDA, M_3, so that $T(M_3) = L_3$.

Proof.

1. M_3 mirrors the actions of M_2 except that when a match for the S^jQ^i from the store is found, reconstruct the store, and look for another match. If none is found, that is if \$ is encountered, then erase the top of the store and proceed as M_2.

2. When it is verified that x defines a complete, minimal description of the transitions, put n(q) Q's on the store and search, in a manner analogous to previous searches, for a substring (q^i), where i is the number of Q's on the store. If there is no substring (q^i), erase the top Q from the store and search with Q^{i-1} on the store. If there is a substring (q^i) when Q^i is on the store, reconstruct the store and search for duplication. If no duplicate is found, continue as if no substring (q^i) were found. When the last Q is erased, x is accepted. Q.E.D.

We are now ready to construct a 2DPDA, U, universal over 1NFA. U will be universal in the following sense. If $M \in$ 1NFA and x is to be submitted to M, we will choose a string y in L_3 describing $M' \in$ 1DFA such that $T(M') = T(M)$. We now submit $yE(x)$ to U, where E(x) is the encoding of string x. If $x = s_{i1}s_{i2}\ldots s_{in}$ where s_{ij} is symbol i_j, then

$E(x) = S^{i1}|S^{i2}|\ldots|S^{in}$. (For example, if $x = s_1s_4s_2s_1$, $E(x) =$ $S|SSSS|SS|S$.) U accepts $yE(x)$ if and only if M accepts x.

THEOREM 1. There exists a 2DPDA, U, which is universal over the class of FA.

Proof.

1. U first checks its input and rejects unless the input consists of a string in L_3 (the description of M') followed by an encoding of a string $x \in \Sigma_M^*$. During the check, U behaves like M_3 except that it interrupts its check of the description when the store contains $\#S^{n(s)}$. At this point, U determines if what follows the description is indeed the encoding of some string, and moreover, if the encoding contains any substring of the form S^i for $i > n(s)$. Such a substring implies that x contains a symbol not in the alphabet of M'. After checking the encoded string, U resumes the check of the description for membership in L_3.

2. U next copies $(E(x))^R$, the reverse of $E(x)$, onto the store. M' is assumed to start in state 1; a Q is placed on the top of the store. If $x = s_{i1}s_{i2}\ldots s_{in}$, $n > 0$, the store now contains $\#S^{in}|\ldots|S^{i2}|S^{i1}Q$ and step 3 follows. If $x = \epsilon$, the store contains #Q and step 4 follows.

3. The store contains $\#\ldots|S^jQ^i$ for some i and j signifying that M' is in state i with its read head scanning symbol j. U now searches for the substring $q^is^j:q^1)$ in the description. The search is performed as in step 4 of Lemma 2 until the match is found. If | is now the top symbol on the store, U erases the |, places Q^1 on the store and performs another search for the matching transition. Each match destroys one

encoded symbol on the store. When # is the top symbol after a match, the input has been completely scanned; U then places Q^1 on the store and step 4 follows.

4. The store now contains $\#Q^i$ for some i; U searches the description for the substring (q^i). If a match is found, state q_i is final and U accepts; otherwise U rejects. Q.E.D.

A 2DPDA Which Accepts a Diagonal Language for 1NFA

A slight modification of U allows us to construct a 2DPDA,D, which accepts a diagonal language for 1NFA.

Let $L_4 = \{x | x \epsilon L_3, x \text{ describes } M, x \notin T(M)\}$.

THEOREM 2. There exists a 2DPDA,D, which accepts L_4.

Proof. We first adopt a convention concerning the five symbols in the alphabet of L_4: q,s,:,(and) will be interpreted as s_1, s_2, s_3, s_4 and s_5, respectively. It is now meaningful to submit a description x of M to M.

1. D first checks x and rejects if $x \notin L_3$.

2. D next searches for a substring qs^5: in x. The absence of such a substring implies that the alphabet of M contains less than five symbols; since) is interpreted as s_5 and x must contain at least one occurrence of), $x \notin T(M)$. If qs^5 is absent, D accepts; otherwise, step 3 follows.

3. D now places an encoding of x^R on the store. The symbols q,s,:,(and) are encoded, S, SS, SSS, SSSS, and SSSSS, respectively; encoded symbols are separated by | on the store. For example, if

x = qs:q)...(q) the store contains $\#S^5|S|S^4|...|S^5|S|S^3|S^2|S$.

4. A Q is placed on the store and D performs Step 3, Theorem 1, like U; this simulates the moves of M until the store contains only $\#Q^i$ for some i.

5. D now searches x for the sbustring (q^i). If present, D rejects since M accepts x; if absent, D accepts since M rejects x. Q.E.D.

THEOREM 3. L_4 is not a Context-Free Language.

Proof. If L_4 is Context-Free, then there exist integers n, m so that every $z \in L_4$ with length (z) > n can be decomposed into a string uvwxy, with $vx \neq \epsilon$, length (vwx) $\leq$ m, and all strings of the form uv^kwx^ky are in L_4 for each $k \geq 1$ [3].

Let $z \in L_4$ be a string such that length (z)>n. Such a string exists since L_3 contains strings of arbitrary length describing DFA with no final states; each such string is in L_4. There exist strings u,v,w, x,y such that z = uvwxy, $vx \neq \epsilon$, and if $k \geq 1$ then $z_k = uv^kwx^ky \in L_4$. The contention is that only $v = x = \epsilon$ can have this property. We show that this is true by the process of elimination.

The nested sets $L_4 \subset L_3 \subset L_2 \subset L_1$ were carefully constructed to facilitate this proof. Repetition of any substring will cause at least one of the following: a) an illegal substring in z_2; b) a repetition of a state transition or final state description in z_2 or z_3; c) a description with n(q) or n(s) increased, in which case an incomplete description results.

Case 1. x begins with q.

a. $x = q^i$ for some $i \geq 1$. $z_k \notin L_2$ if $k > n(q)$.

b. $x \neq q^i$ for any $i \geq 1$. $z_2 \notin L_1$ or $z_3 \notin L_3$.

Case 2. x begins with s.

a. $x = s^i$ for some $i \geq 1$. $z_k \notin L_2$ if $k > n(s)$.

b. $x \neq s^i$ for any $i \geq 1$. $z_2 \notin L_1$ or $z_3 \notin L_3$.

Case 3. x begins with :. $z_2 \notin L_1$ or $z_2 \notin L_3$.

Case 4. x begins with (. $z_2 \notin L_1$ or $z_2 \notin L_3$.

Case 5. x begins with). $z_2 \notin L_1$ or $z_3 \notin L_3$.

The argument for v is identical; thus $v = x = \epsilon$. Q.E.D.

Universal 2DPDA(h) and 2NPDA(h)

In Theorem 4 we show that, for each $h \geq 1$, 2DPDA(h) contains a member U which is universal over the class. Lemmas 4, 5, and 6 show that a machine universal over 2DPDA(h) need only be universal over R(h), the restricted subclass of members with three-symbol store and four-symbol input alphabets; the proof of Theorem 4 takes advantage of this fact.

LEMMA 4. Let $h \geq 1$. For each $M \in$ 2DPDA(h) there exists an equivalent machine $M' \in$ 2DPDA(h) with $\Gamma_{M'} = \{\#, Z, |\}$.

Proof. If $|\Gamma_M| \leq 3$ then the result is trivial. Assume that $\Gamma_M = \{Z_i | 0 \leq i \leq k\}$; the bottom of M's store is Z_0. For each $1 \leq i \leq k$ assign the codeword $|Z^i$ to Z_i; assign # to Z_0. M can easily be modified so that it codes symbols of Γ_M into $\{\#, Z, |\}$ when it writes on the store and

decodes them when it reads from the store. There is a special symbol for Z_0 so that there will be no possibility of M' emptying its store while decoding a symbol. The details of the proof are omitted; results such as this are well known for similar devices [12,19]. Q.E.D.

Lemma 4 shows that the subclass of machines with a three-symbol store alphabet spans the class 2DPDA(h). Such a strong result is not available for the subclass of machines with a four-symbol input alphabet. We show in Lemma 5, however, that this subclass spans, within an input encoding, all of 2DPDA(h). This closely parallels techniques used to construct universal TM [12,14]. We define the input encoding function I as follows. Given any string $x = \text{¢}s_{i_1}s_{i_2}\ldots s_{i_n}\$$, where s_{i_j} is symbol $i_j \in \Sigma_M - \{\text{¢},\$\}$, I(x) is $\text{¢}'S^{i_1}|S^{i_2}|\ldots|S^{i_n}\$'$.

LEMMA 5. Let $h \geq 1$. For each $M \in$ 2DPDA(h) there exists $M' \in$ 2DPDA(h) with $\Sigma_{M'} = \{\text{¢}',\$',S,|\}$ such that M accepts x if and only if M' accepts I(x).

Proof. Since each machine in 2DPDA(h) has a finite input alphabet, M' can easily decode symbols before making any move. Since ¢' and $' are not encoded, there is no possibility that M' will leave its input while decoding the symbols. Q.E.D.

LEMMA 6. Let $h \geq 1$. If U is a machine which is universal over R(h), then U is universal over 2DPDA(h).

Proof. If x is to be submitted to $M \in$ 2DPDA(h), we first apply

Lemma 4 and obtain an equivalent machine M' with a three-symbol store alphabet. We next apply Lemma 5 and obtain a machine M'' with a four-symbol store alphabet such that $x \in T(M')$ if and only if $I(x) \in T(M'')$. The string $\alpha_{M''}E_{M''}(I(x))$ is now submitted to U. U accepts $\alpha_{M''}E_{M''}(I(x))$ if and only if M'' accepts I(x) which occurs if and only if $x \in T(M') = T(M)$. Since the composite function $E_{M''}(I)$ is an encoding function, U is universal over all of 2DPDA(h). Q.E.D.

We now define the language, L_{1_k}, of R(k) descriptions. A string x over {q,¢',$',z,s, ,#,:,(,),+,0,-} is in L_{1_k} if and only if it is a finite concatenation of substrings of one of the following types.

1. $s_1s_2\ldots s_kZ_1q^j:h_1h_2\ldots h_kq^{j'}Z_1Z_2$.
2. $s_1s_2\ldots s_kZ_1q^j:h_1h_2\ldots h_kq^{j'}Z_1$.
3. $s_1s_2\ldots s_kZ_1q^j:h_1h_2\ldots h_kq^{j'}$.
4. (q^i).

Here $h_i \in \{+,0,-\}$, $1 \le i \le k$, $Z_1,Z_2 \in \{\#,Z,|\}$, $s_i \in \{¢',\$',S,|\}$, $1 \le i \le k$, and j,j' positive integers.

Type 1, 2, and 3 substrings represent an allowed transition from state j to state j' when symbols $s_1,s_2,\ldots,s_k$ are scanned by heads 1,2, ...,k, respectively, and symbol Z_1 is on top of the store. After appropriate store action, i.e. adding symbol Z_2 to the store in type 1, leaving the store unchanged in type 2, or erasing symbol Z_1 in type 3, head p is moved in the direction indicated by h_p for $1 \le p \le k$. A type 4 substring indicates that state i is a final state. L_{1_k} is clearly regular for each $k \ge 1$. More than one substring beginning with $s_1s_2\ldots s_kZ_1q^j:$ indicates possible nondeterminism in the automaton's description; any

member of L_{1_k} can be a description of a deterministic machine if we assume that the left-most applicable substring is always chosen. Additionally, we agree that if M is described by a member of L_{1_k}, then M begins in state 1 with # on the store.

If machine $M \in R(k)$ is to be simulated and M has n(q) different states, then inputs to M will be encoded by the encoding function E_M. Let g_M be the 1-1 function from $\Sigma_M = \{¢', \$', S, |\}$ to $(\Sigma_M \cup \{\beta\})^*$ such that $g_M(t) = t\beta^{n(s)}$ for each $t \in \Sigma_M$. Then for $t \in \Sigma_M$, $w \in \Sigma_M^*$ we define $E_M(t) = g_M(t)$ and $E_M(tw) = g_M(t)*E_M(w)$. Thus if $x = ¢'SS|S\$'$ and M has four states, $E_M(x)$ is $¢'\beta\beta\beta\beta*S\beta\beta\beta\beta*S\beta\beta\beta\beta*|\beta\beta\beta\beta*S\beta\beta\beta\beta*\$'\beta\beta\beta\beta$.

THEOREM 4. For each $k \geq 1$, there exists $U \in 2DPDA(k)$, U universal over 2DPDA(k).

Proof. We will only show that U is universal over R(k). U must first check that its input $\alpha_M E_M(x)$ is syntatically correct. Only one head is needed for this check, $\alpha_M \in L_{1_K}$, a regular language; this portion can be easily checked. U next checks that what follows α_M is indeed the encoding E_M of a string $x = ¢'y\$'$, $y \in \{S, |\}^*$; this requires checking for the proper number of β's following each symbol of x and for proper placement of the separator *. U finds n(q), the number of states of M, from the longest substring of q's in α_M and puts a unary count of this length on the store. The check for β's proceeds similar to the check in Lemma 3; other aspects of the syntax check are trivial.

The bottom element of U's store is @. U's store is used to remember the store of M, position of read head 1 of M and M's current

state. If U's store is

$$@\#Z_1Z_2\ldots Z_p1^mQ^j,\ Z_i \in \{|,Z\} \quad 1 \le i \le p,$$

then M's store contains $\#Z_1Z_2\ldots Z_p$, head 1 of M is scanning the m*th* symbol in M's input and M is in state j.

Heads 2 through k of U will monitor the position of heads 2 through k of M, normally resting on the symbol of Σ_M they scan. Head 1 of U will search α_M for allowable moves of M.

Phase (i) *Initialization.* If k > 1, U moves heads 2 through k to the first symbol in $E_M(x)($ ¢' $)$, #1Q is added to U's store. U remembers in its control the symbol scanned by each of head's 1 through k (a ¢') and the top symbol of M's store (a #). Simulation begins as U enters phase (ii).

Phase (ii) *Search for Move.* U uses head 1 to search α_M left-to-right for an applicable move. If symbols s_j, $1 \le j \le k$, were scanned by the k heads of M, respectively, and symbol Z_p is on M's store, U must locate a substring in α_M which begins $s_1s_2\ldots s_kZ_p$. If no such substring can be found U rejects the input since M has no allowed move from this configuration. If such a substring is found, U next checks that the state depicted on the top of the store matches the state in the next portion of the substring. Store reconstruction and resumed search must follow unsuccessful matches. If U can find an applicable substring, it enters phase (iii); otherwise, U rejects.

Phase (iii) *Update Head Position.* When U has found a move for

M, U will have erased the Q's from its store and a : will be scanned by head 1. Head 1 is moved one square to the right where it now scans the move for head 1 of M. At this time, the store of M contains

$$@\#Z_1Z_2\ldots Z_p1^m$$

U adds a 1, erases a 1 or does nothing to its store according to whether the symbol under head 1 is +, - or 0, respectively. If the top of U's store is not a 1 after this modification, then M has moved head 1 off of the left end of its input and U halts in a non-accepting state. Otherwise, U moves head 1 one square right and begins to update the positions of head 2 if $k > 1$ or enters Phase (iv) if $k = 1$.

If head 1 is currently scanning a 0, head 2 does not move. If head 1 is scanning a +, head 2 moves right past the β's, until it scans a * or $. If this symbol is a $, head 2 has moved off the right of M's input and U rejects. Otherwise, head 2 moves one square right to the next symbol of M's input. If head 1 is scanning a -, head 2 moves one square left. If head 2 is now scanning a *, it moves left past the β's to the next ¢',S or | it encounters; otherwise head 2 has moved off the left of M's input and U rejects. The updating of head 2 is now complete and head 1 moves right one square. If $k > 2$, the updating of heads 3,...,k proceeds in the same manner. When all heads have been updated, head 1 of U will be scanning the left end of a substring of q's which depict the new state of M. Head 1 moves right through this substring, adding a Q to the store for each q scanned. When head 1 is no longer

scanning a q, the new state of M has been placed on the store of U. Head 1 now reads, and causes U to remember in its control, the modifications which U must make to the store of M.

At this time, the store of U contains

$$@\#Z_1Z_2\ldots Z_p 1^{m'}Q^{j'}$$

U will use the count m' to position head 1 on the proper symbol of M's input. Although U knows the modifications it must make to M's store, it has not as yet made these modifications. These adjustments are made in phases (iv) and (v). U now enters phase (iv).

Phase (iv) *Get New Symbol for Head 1.* Head 1 is moved right to the ¢' in $E_M(x)$. For each Q on the store, it moves one square right (into the string of β's) and erases the Q. When all Q's have been erased, head is located j' β's from the last symbol in M's input; the store is devoid of Q's. The top of the store is erased, and if the new top is not a 1, U enters phase (v). Otherwise, head 1 moves left adding a Q to the store for each left move until it no longer scans a β. The j' Q's have been replaced; head 1 now moves right to the first S,|,$ or $'. The process described above for the first symbol in M's input is repeated if head 1 is not scanning $, U entering phase (v) when all 1's are erased from its store. If the m' 1's are not removed before U reaches $, then head 1 of M has left the right side of its input and U rejects.

Phase (v) *Update Store and Replace Count and* $Q^{j'}$. When phase

(v) is entered, head 1 of U is located j' β's from the symbol to be scanned by head 1 of M. The store of U contains

$$@\#Z_1Z_2\ldots Z_p$$

U can now modify M's store with the update information which it has remembered in the finite control. U adds a 1 to the store as a first step in replacing the m' 1's which denote the position of head 1 of M. Head 1 is now moved left until it no longer scans a β, adding a Q to the store for each left move. Head 1 of U is now scanning the symbol that head 1 of M is scanning. This symbol is remembered in the finite control. If this symbol is not ¢', the count used to locate head 1 of M has not been replaced. Head 1 is moved left one square to the *, left until it no longer scans a β, and then right one square (into the string of β's) erasing a Q from the store for each right move until a 1 is the stop of the store. U now adds a 1 to the store and replaces the Q's as before. If this symbol is not ¢', U repeats the process described above; if it is a ¢', the m' 1's have been replaced, as have the j' Q's. U's store contains

$$@\#Z_1Z_2\ldots Z_p,1^{m'}Q^{j'}$$

Phase (vi) is entered now.

Phase (vi) *Final State Check.* The symbol scanned by head 1 of M is already in the finite control. For $k > 1$, heads 2,...,k of U are

scanning the symbols scanned by the corresponding heads of M and are now remembered in the finite control. If every head of M is now scanning the \$', U begins a final state check using head 1 to search α_M as in Theorem 2. If M is in a final state, U accepts; if M is not in a final state or if all heads of M were not scanning \$', U moves head 1 left to ¢ and reenters Phase (ii).

By Lemma 6, the proof is complete. Q.E.D.

COROLLARY 4.1. For each $h \geq 1$, there exists $U \in$ 2DPDA(h), U universal over 2DFA(h).

Proof. This is obvious since 2DFA(h) $\subseteq$ 2DPDA(h). Q.E.D.

THEOREM 5. For each $h \geq 1$, there exists $U' \in$ 2NPDA(h), U' universal over 2NPDA(h).

Proof. The results of Lemmas 4, 5, and 6 are easily seen to hold for 2NPDA(h). The results analogous to Lemmas 4 and 5 follow from the observation that a nondeterministic machine decodes store and input symbols deterministically and then selects its move nondeterministically; the result analogous to Lemma 6 requires no further comment.

U' searches α_M for an applicable move just like U of Theorem 4; when it finds a move it nondeterministically decides whether to simulate the move like U or search α_M to the right for another applicable move. If U' elects to search for another move, head 1 moves left through the q's, replacing a Q on the store for each q read, and then moves right to continue the search. Q.E.D.

COROLLARY 5.1. For each $h \geq 1$, there exists $U \in$ 2NPDA(h), U universal over 2NFA(h).

Proof. This is obvious since 2NFA(h) $\subseteq$ 2NPDA(h). Q.E.D.

LEMMA 7. Let $h \geq 2$. For each $M \in$ 1DFA(h) there exists $\bar{M} \in$ 2DPDA(h-1) such that $T(M) = T(\bar{M})$.

Proof. For each input x, $\bar{M}$ uses head 1 to copy x^R onto its store. When this is done, all of $\bar{M}$'s heads will be scanning ¢. Heads 1 through h-1 of $\bar{M}$ will correspond to heads 1 through h-1 of M, respectively. The symbol scanned by head h of M will be on the top of $\bar{M}$'s store. If head h of M moves right $\bar{M}$ erases the top symbol on the store; otherwise $\bar{M}$ makes no modification of the store. The finite control of $\bar{M}$ will exactly correspond to M's control except that $\bar{M}$ looks on its store rather than in its input for the symbol scanned by head h of M. Q.E.D.

THEOREM 6. For each $h \geq 2$, there exists $U \in$ 2DPDA(h-1) and $U' \in$ 2NPDA(h-1), U universal over 1DFA(h) and U' universal over 1NFA(h).

Proof. The existence of U follows immediately from Lemma 7 and Theorem 4.

It is easy to see that the results of Lemma 7 hold for $M \in$ 1NFA(h) and $\bar{M} \in$ 2NPDA(h-1); M simply moves nondeterministically like M. The existence of U' now follows immediately from Theorem 5. Q.E.D.

LEMMA 8. For each $h \geq 1$, 2DPDA(h) cannot contain a member

universal over 2DPDA(h+1) and 2NPDA(h) cannot contain a member universal over 2NPDA(h+1).

Proof. Let $h \geq 1$. Ibarra [13] has shown that 2DPDA(h) $\subset$ 2DPDA(h+1) and 2NPDA(h) $\subset$ 2NPDA(h+1). Let L be a language such that $L = T(M)$ for some $M \in$ 2DPDA(h+1) but L is not accepted by any member of 2DPDA(h). Suppose $U \in$ 2DPDA(h) and U is universal over 2DPDA(h+1). There is a word α_M and an encoding E_M such that U accepts $\alpha_M E_M(x)$ if and only if M accepts x. From U we now show how to construct a member $M' \in$ 2DPDA(h) which accepts x if and only if U accepts $\alpha_M E_M(x)$. M' with input ¢x$ simulates the action of U with input ¢$\alpha_M E_M(x)$$ as follows:

1. States of M' will be used to encode the string α_M of length p and the position of each head of U relative to α_M, i.e. to the left of α_M, to the right of α_M, or scanning the symbol in position i of α_M, $1 \leq i \leq p$. Head motion in U is "remembered" through states in M' in an obvious manner. Head j of M' is positioned at the ¢ if head j of U is to the left of or in α_M. Head j of M' moves right if head j of U moves off α_M to the right into $E_M(x) = E_M(t_1 t_2 \ldots t_n) = g_M(t_1)|g_M(t_2)|\ldots|g_M(t_n)$.

2. When head j of M' moves right to scan a symbol $t \neq \$$, states of M' are used to encode $g_M(t)$ and the position of head j of U, initially at the left within $g_M(t)$. When head j of M' moves left to scan a symbol $t \neq ¢$, the encoding is identical but the position is initially at the right of $g_M(t)$. If head j of U moves off $g_M(t)$, it now scans |, $, or the right-most symbol of α_M. Head j of M' can easily determine which case occurs; M' will move head j accordingly unless the | must be created in the control. In this case M' must delay motion until head j of

U moves left off $|g_M(t)$ or right of $g_M(t)|$.

Further details of the simulation should be obvious.

We have now constructed M' ϵ 2DPDA(h) such that T(M') = $\{x|\alpha_M E_M(x) \epsilon T(U)\}$ = T(M) = L, a contradiction.

The same argument is valid for 2NPDA(h) and 2NPDA(h+1). Q.E.D.

THEOREM 7. For each h $\geq$ 1, 1DFA(h) $\subset$ 2DPDA(h) and 1NFA(h) $\subset$ 2NPDA(h).

Proof. Let h $\geq$ 2. Clearly, 1DFA(h) $\subseteq$ 2DPDA(h); if 1DPA(h) = 2DPDA(h), then by Theorem 6 there exists U ϵ 2DPDA(h-1) such that U is universal over 1DFA(h) = 2DPDA(h). This contradicts Lemma 8. An analogous contradiction arises if 1NFA(h) = 2NPDA(h). For the case h = 1, it is already known [14] that 1DFA = 1NFA $\subset$ 1DPDA $\subset$ 2DPDA $\subset$ 2NPDA. Q.E.D.

Universal 2DFA(h) and 2NFA(h)

In Theorem 8 we show that, for each h $\geq$ 1, 2DFA(h+1) contains a member, U, universal over 2DFA(h).

We first show, in Lemma 9, that U need only be universal over R'(h), the restricted subclass of 2DFA(h) whose members have input alphabet {¢',$',S,|}. R'(h) is analogous to R(h); the restricted subclass of 2DPDA(h) whose members have this input alphabet and a three-symbol store alphabet.

LEMMA 9. Let h $\geq$ 1. If U is a machine universal over R'(h), then U is universal over 2DFA(h).

Proof. The proof is similar to the proofs of Lemmas 5 and 6 and uses the same encoding I. Let $h \geq 1$. If $M \in 2DFA(h)$, there exists $M' \in R'(h)$ such that $x \in T(M)$ if and only if $I(x) \in T(M')$. The existence of M' follows when the proof of Lemma 5 is applied to 2DFA(h) rather than 2DPDA(h). U accepts $\alpha_{M'}E_{M'}(I(x))$ if and only if M' accepts I(x) which occurs if and only if M accepts x. Since $E_{M'}(I)$ is an encoding function, U is universal over all of 2DFA(h). Q.E.D.

We begin by describing the regular language, L_{2_k}, of R'(k) descriptions. A string x is in L_{2_k} if and only if it is a finite concatenation of substrings of one of the following types.

1. $q^j s_1 s_2 \ldots s_k : h_1 h_2 \ldots h_k q^{j'})$.
2. (q^j).

A type 1 substring represents an allowed transition from state j to state j' when symbols $s_1, s_2, \ldots, s_k$ are scanned by heads $1,2,\ldots,k$, respectively. A type 2 substring indicates that state j is a final state. Strings in L_{2_k} will depict a deterministic FA if it is agreed that the left-most applicable move is the only valid one. Machines described by members of L_{2_k} start in state 1. The encoding E_M described for $R(k) \subseteq 2DPDA(k)$ will also be used for FA.

THEOREM 8. For each $k \geq 1$, there exists $U \in 2DFA(k+1)$, U universal over 2DFA(k).

Proof. We will only show that U is universal over R'(k). U must first verify that its input, y, is of the form $\alpha_M E_M(x)$. Since L_{2_k} is regular, the first part of the task is trivial. U next checks to see if

what follows α_M is indeed the encoding E_M of a string $x = ¢'t\$'$, $t \in \{S,|\}^*$. Only one head is required to verify that each symbol in x is followed by a string of β's and that * separates the encoded symbols; however, a second head is needed to decide whether the proper number of β's follows each symbol in x. U must determine n(q) and compare the length of each string of β's to n(q). The universal member of 2DPDA(k) used its store to determine, record, and compare n(q) with the strings of β's. Lacking a store, U will use the position of head 1 from the ¢ in place of a count on the store. Left movement corresponds to store erasure, right-movement to store addition, scanning ¢ to finding # on top of the store and head repositioning to store reconstruction. U rejects if y is not syntactically correct; otherwise, U enters phase (i).

During simulation, head 1 of U will monitor, in $E_M(x)$, the current state of M as well as the position in x of M's head 1. If M is in state j head 1 of U will normally scan the *jth* β to the right of the corresponding ¢', $', S, or | in $E_M(x)$. Heads 2 through k of U will monitor, in $E_M(x)$, the positions in x of M's heads 2 through k, respectively. These heads of U will normally scan the corresponding ¢', $', S, or | in $E_M(x)$. Head k + 1 of U will search α_M for moves of M.

Phase (i) *Initialization.* U starts by moving heads 2 through k to ¢' and head 1 to the first β to the right of ¢'. M is now in state 1. Head k + 1 remains at the ¢. Phase (ii) is entered now.

Phase (ii) *Search for Move.* Head k + 1 moves right to the first type 1 substring. U compares the number of q's at the beginning of the substring with position of head 1 in the β's (the current state of M);

this comparison is similar to the one used above to check that n(q) β's followed each encoded symbol. Here, however, ¢', $', S, or | replaces ¢ in the positioning. Head 1 is repositioned in the β's following each failure, and phase (ii) is reentered. When a match is found head 1 of U is scanning the ¢', $', S, or | scanned by head 1 of M. Head k + 1 now moves right through $s_1 s_2 \ldots s_k$ in the substring, checking that s_i is being scanned by head i, $1 \leq i \leq k$. If any head is not scanning the correct symbol, head 1 is repositioned in the β's and phase (ii) is reentered. If at any time head k + 1 moves off α_M, this means that no applicable move exists; U then rejects.

When an applicable substring is found, U enters phase (iii) with head k + 1 scanning the : in the type 1 substring and heads 1 through k scanning their respective ¢', $', S, or |.

Phase (iii) *Update.* Head k + 1 moves right one square and updates the position of head 1. If head k + 1 scans a + (-), head 1 is moved right (left) to the next ¢', $', S, or |. If such a symbol cannot be found, U rejects since M has moved off its input. If head k + 1 scans a 0, head 1 is not moved.

For i=2,...,k, head i is updated in the same manner. After all k heads have been updated, U remembers if all k heads were scanning $'. Head k + 1 is now scanning the last +, -, or 0 used in updating head k. U must now record the new state of M by correctly positioning head 1. U moves head k + 1 right into the q's, moving head 1 right one square into the β's for each q encountered. When head k + 1 scans), head 1 is properly positioned. If all k heads of M were scanning $', phase (iv)

is entered; otherwise, phase (ii) is reentered.

Phase (iv) *Final State Check.* Head k + 1 now searches α_M left-to-right for a type 2 substring whose number of q's matches the position of head 1 in the β's. Head 1 repositioning occurs after each non-match, and the search continues to the right. If no match is found when head k + 1 leaves α_M, the state is non-final and phase (ii) is reentered. If a match is found, the state is final and U accepts.

By Lemma 9, the proof is complete. Q.E.D.

THEOREM 9. For each $k \geq 1$, there exists $U' \in 2NFA(k+1)$, U' universal over 2NFA(k).

Proof. U' operates just like U in Theorem 8, except that when a move is found, U' non-deterministically decides whether to accept the move or to reposition head k + 1 and search right for another move. Q.E.D.

A 2WPDA Universal Over 2NPDA(h) For All h

In Theorem 10 we construct a member $U \in 2WPDA$, U universal over the class $\bigcup_{k=1}^{\infty} 2DPDA(k)$. Lemmas 4, 5 and 6 indicate that the universal machine needs only to be universal over $\bigcup_{k=1}^{\infty} R(k)$. A string $\alpha_M \in L_{3_k}$ will be a description of a machine $M \in R(k)$. The language $L = \bigcup_{k=1}^{\infty} L_{3_k}$ is the language of description for the class $\bigcup_{k=1}^{\infty} R(k)$. A string is in L_{3_k} if and only if it is a finite concatenation of substrings of one of the following types.

1. $s_1 s_2 \ldots s_k q^j Z_1 : Z_1 Z_2 q^{j'} h_1 h_2 \ldots h_k$.

2. $s_1 s_2 \ldots s_k q^j Z_1 : Z_1 q^{j'} h_1 h_2 \ldots h_k$.

3. $s_1 s_2 \ldots s_k q^j Z_1 : q^{j'} h_1 h_2 \ldots h_k$.

4. (q^j).

These substrings correspond exactly to substrings in L_{1_k}; the order of symbols in a substring has been slightly changed to facilitate the proof of Theorem 10. The encoding E_M is the same as in Theorem 4 except for the number of β's; this number is now k for $M \in R(k)$. We will denote this encoding E_k, since it is the same for all $M \in R(k)$.

THEOREM 10. There exists $U \in$ 2WPDA, U universal over $\bigcup_{h=1}^{\infty}$ 2DPDA(h).

Proof. We will show that U is universal over $\bigcup_{h=1}^{\infty} R(h)$. In simulating $M \in R(k)$, U will operate similarly to U of Theorem 4. Differences occur when updating the positions of the heads; the writing capacity of U will only be used to make this modification. A % instead of a β in the *ith* space following an input symbol means that head i of M is scanning that symbol. The bottom of U's store is @; if the store of U is

$$@\#Z_1 Z_2 \ldots Z_p Q^j S_k S_{k-1} \ldots S_1$$

then M is in state j, head i, $1 \le i \le k$, is scanning $S_i \in \{¢', \$', S, |\}$ and the store contains $Z_1 Z_2 \ldots Z_p$, $Z_j \in \{Z, |\}$, $1 \le j \le p$.

1. U must first verify that the input string, y, submitted to it is of the form $\alpha_M E_M(x)$, where α_M is in L_{3_k}. Since descriptions of pushdown automata with any number of heads can be submitted to U, U must

first search the α_M portion of y for a type 1, type 2, or type 3 substring in order to get a value for k, a unary count of which is put onto the store of U. (If there are no type 1, 2 or 3 substrings in α_M, then α_M describes the null automaton and y is rejected since no strings are accepted by a null automaton.) U uses the proposed value of k on its store to verify that the portions $S_1S_2\ldots S_p$ and $h_1h_2\ldots h_r$ of each type 1, 2 and 3 substring have $p = r = k$; store reconstruction follows each successful test. Other details of the syntax test for membership in L_{3_k} are omitted. Next, U verifies that the remainder of y is indeed in $E_k(x)$. Since k 1's are on the store, this test can be performed as in Theorem 4, except that the k 1's replace the n(q) 1's. U rejects if y is not of the proper form.

2. U begins simulation by erasing the k 1's, placing a #Q on the store and writing a % over each β in the encoded ¢' of M; a ¢' is added to the store for each %.

3. Since k is arbitrary, U cannot use the finite control to "remember" the k-tuple of input symbols under the k heads of M. If the store of U is $@\#Z_1Z_2\ldots Z_pQ^jS_k\ldots S_1$, then U searches α_M for a substring which begins with $s_1s_2\ldots s_kq^jZ_p:$. The search is exactly like that of Theorem 1, reconstruction following unsuccessful matches. If there is no such substring in α_M, U rejects; otherwise, U will move one square right (past the :) and copy the store modifications and new state of M onto its store.

4. When U encounters the head movement portion of the substring, i.e. $h_1h_2\ldots h_k$, the head is moved right to the first symbol not +, - or

0; a 1 is added to the store with right move. Head 1 then moves left one space to h_k, and h_k is changed to +', -' or 0' according to whether h_k was +, -, or 0. The store now has k 1's on top of its previous contents. U moves right through $E_k(x)$, using the count on the store to locate the encoded symbol with a % in this space. The store, now empty of 1's, is reconstructed by moving left to the first symbol in {¢',$',S |}, adding a 1 to the store for each move. The search continues until the % is found, at which time it is overwritten with a β. U now restores the count to the store. In order to accomplish correct head movement at this time, three distinct sets of states were used for the above operations according to whether h_k was a +, - or 0. If h_k was + (-) the head is moved right (left) to the next symbol in {¢',$',S,|}. If h_k was 0 the head remains stationary. A % is now placed in the k*th* space to the right of this symbol; the store is used to locate this space. Again, the store is devoid of 1's. The head now moves left to the symbol itself and copies it onto the store. The upper portion of the store is now the new symbol being scanned by head k.

The head now moves left to the first +', -', or 0' it finds (it is the only one), changes it back to +, -, or 0, respectively, and moves left one space. If the new symbol is +, -, or 0, U has not yet updated all k heads and the process described above for simulating the move of head k and copying its scanned symbol onto the store is repeated for head k - i, $1 \leq i \leq k-1$. After updating head 1, the symbol to the left of the +, -, or 0 will not be +, -, or 0.

Note that if any of the k heads leave x in M, U will search

outside of $E_k(x)$ for a non-existent symbol, eventually leaving ¢y$ and rejecting y.

5. If any head of M is not scanning its $' (easily detected if any space in the encoded $' contains a β) step 3 is entered. However, if all k heads of M are scanning the $', all $' are removed from the store and the configuration is tested for finality and acceptance of y as in Theorem 4. A non-final configuration, however, requires an essentially different method of store reconstruction since k is arbitrary. U moves right to the first % in the encoded $' and replaces a $' on the store for each % read. Step 3 is then entered.

By Lemma 6 the proof is complete. Q.E.D.

COROLLARY 10.1. U is also universal over $\bigcup_{h=1}^{\infty}$ 2NPDA(h).

Proof. Since Cook [5] has shown that $\bigcup_{h=1}^{\infty}$ 2DPDA(h) = $\bigcup_{h=1}^{\infty}$ 2NPDA(h), this result is a trivial consequence of Theorem 10.

COROLLARY 10.2. T(U) is not accepted by any M ∈ $\bigcup_{h=1}^{\infty}$ 2NPDA(h).

Proof. If there is an h ≥ 1 and M ∈ 2NPDA(h) such that T(M) = T(U), then M is universal over $\bigcup_{h=1}^{\infty}$ 2NPDA(h) and, consequently, 2NPDA(h+1). This contradicts Lemma 8. Q.E.D.

CHAPTER III

CERTAIN CFL, 2DPDA(2) AND 2DPDA(3)

Notation

A context-free grammar G consists of a non-terminal alphabet v_N, a terminal alphabet v_T, a unique start symbol $S \in v_N$ and a set of production rules P with $\alpha \to \beta \in P$ if and only if $\alpha \in v_N$, $\beta \in (v_N \cup v_T)^*$. We will use upper case letters to denote elements in v_N, lower case letters for elements for v_T. If $\alpha \to \beta$ is in P, $\gamma,\delta \in v_T^*$ then the string $\gamma\beta\delta$ can be derived from $\gamma\alpha\delta$ by using production $\alpha \to \beta$, i.e. $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$. If $\alpha_1,\alpha_2,\ldots,\alpha_n$ are in $(v_T \cup v_N)^*$ and $\alpha_1 \Rightarrow \alpha_2 \ldots \Rightarrow \alpha_n$, we write $\alpha_1 \overset{*}{\Rightarrow} \alpha_n$, i.e. α_n is derived from α_1. We can now define L(G), the language of the grammar G. $L(G) = \{w \mid w \in v_T^* \text{ and } S \overset{*}{\Rightarrow} w\}$.

Minimal Linear Languages and 2DPDA(2)

A context-free grammar, G, is a minimal linear grammar [9] provided $v_N = \{S\}$ and the production rules of G are of one of the following two types:

(i) $S \to \alpha S \beta$
$\alpha,\beta \in (v_T - \{c\})^*$

(ii) $S \to c$.

A minimal linear grammar generates terminal strings from the ends toward the special symbol c. We will show how to construct a machine $M \in$ 2DPDA(2) which uses the two read heads to parse candidate strings in the same way. The store of M will be used to keep a record of which rules of G have been used in the parse. The input alphabet of M will be

$V_T \cup \{¢,\$\}(¢,\$ \notin V_T)$. If x is to be tested for membership in L(G), the string ¢x$ will be submitted to M. If G has N type (i) productions, the pushdown alphabet of M will be N + 1 distinct symbols, Z_i, $0 \leq i \leq N$. Z_0 is the initial element of the store. The states of M correspond approximately to the productions of G.

THEOREM 11. For each minimal linear grammar G, a machine $M \in$ 2DPDA(2) can be constructed with T(M) = L(G).

Proof. If x is indeed in L(G), then there is some sequence of productions which generate x. The process described below allows M to test every possible sequence; M rejects x only if there is no sequence of productions which generate x. If x is ambiguous M will generate only one parsing of it. (However, M could be easily modified to generate alternative parsings.) M will halt for all inputs.

1. M moves head 2 right to $ checking that there is exactly one c in the input. If there is not exactly one c in x, M rejects x; otherwise, M moves head 2 one square left and head 1 one square right.

2. If heads 1 and 2 are both scanning c, M accepts x and halts. Otherwise M enters step 3 with I = 1.

3. The type (i) rules of G will be sequentially tested for applicability to x beginning with rule I. Suppose rule I is of the form $S \rightarrow a_{I_1} a_{I_2} \ldots a_{I_m} S b_{I_p} \ldots b_{I_1}$. If rule I could be used then head 1 must be scanning symbol a_{I_1} at the left end of a substring $a_{I_1} a_{I_2} \ldots a_{I_m}$ and head 2 must be scanning b_{I_1} at the right end of a substring $b_{I_p} \ldots b_{I_1}$. M tests rule I for applicability in the following manner. For each

$1 \leq j \leq I_m$, M: (i) verifies that the symbol scanned by head 1 is a_{I_j} (ii) moves head 1 one square to the right. This process is repeated for the "b-portion" of the rule using head 2 which moves left upon a match. If both heads were successful in matching all terminal components of rule I, a Z_I is added to the store and M enters step 2. If an input symbol fails to match the appropriate component of rule I at some point during the test, M can reposition the heads to their positions when step 2 was entered. (This is possible since M can remember in the finite control which rule was being tested and how far it had successfully matched.) If $I < N$, M reenters step 3 with $I = I + 1$; otherwise, M enters step 4.

4. This step is entered when no rule can be applied in step 3. This means that even though every rule selected in step 3 could be applied, the sequence chosen so far cannot generate x. If the top of M's store is Z_i, and $i = 0$, M halts rejecting x since no sequence of productions can generate x; otherwise, M uses rule i to reposition heads 1 and 2 their positions when this rule was selected. M now erases Z_i, sets $I = i + 1$ and enters step 3. Q.E.D.

Linear Languages and 2DPDA(2)

A context-free grammar G will be a linear grammar [2] provided every production of G is of one of the following two types:

(i) $A \rightarrow \alpha B \beta$
$\alpha, \beta \in V_T^*$; $A, B \in V_N$

(ii) $A \rightarrow \alpha$

Observe that if γ is any string derived from S, then γ has at most one non-terminal symbol. It is also true that strings in the

language of a linear grammar are produced from the ends toward the middle. We will show how to construct, for each linear grammar G, $M \in 2DPDA(2)$ with $T(M) = L(G)$. M will work similarly to the machine M of Theorem 11, with modifications to account for the absence of a center marker and the possibility of different non-terminals in a derivation.

THEOREM 12. For each linear grammar G, a machine $M \in 2DPDA(2)$ can be constructed with $T(M) = L(G)$.

Proof. The productions of G are numbered with all type (ii) productions first, ϵ-productions in the first of this list. Note that M can remember, in the finite control, the non-terminal that it currently expanding. It is able, therefore, to choose only those rules whose left side is the same as the current non-terminal in step 2.

1. M moves head 2 right to \$; then head 1 moves one square right and head 2 one square left. I is set to 1.

2. M searches the productions (beginning with rule I) for one that can be applied as in step 3, Theorem 11. Type (ii) productions are handled by having M move head 2 left for each match; head 1 remains stationary throughout. Neither head moves in an ϵ-production. If no rule can be applied, M enters step 4; otherwise, M enters step 3.

3. This step is entered at the completion of every successful rule application. M moves both heads left, adding a * to the store for each left move, until either head is scanning the ¢. If head 1 reaches ¢ before or at the same time as head 2, then the parse is not complete. The count on the store (in *'s) is used to reposition the heads, and if

there was a non-terminal to expand, step 2 is entered with I = 1; if there was no non-terminal to expand, step 4 is entered.

If head 2 scans a ¢ before head 1, it is possible that the parse is complete. The entire string will have been examined if head 1 is one square to the right of ¢. Accordingly, M moves head 1 one square to the left; if head 1 is not scanning ¢, then the heads have "crossed" (i.e. the same symbol has been used by both heads in some rule choice). The heads are repositioned and step 4 is entered. If head 1 did scan ¢, then the parse is complete if either (i) there is no non-terminal left to expand or (ii) the non-terminal left, say B, has a production of the form $B \rightarrow \epsilon$ in G. In either case M halts and accepts x; otherwise, M enters step 4. Note that step 3 must be applied even if the non-terminal left to be expanded does not have an ϵ-production since M must check to see whether the heads have "crossed."

4. This step is like step 4 of Theorem 11, except that in addition to repositioning the heads based upon rule i for Z_i on the store, M must also remember the left side of rule i since this will be the new non-terminal to be expanded. Z_i is erased, I is set to i + 1 and step 2 is reentered (unless of course, i was 0, in which case M halts and rejects x.) Q.E.D.

k-Linear Languages and 2DPDA(3)

A context-free grammar G is a k-linear grammar if all of its productions are of one of the following types [25]:

(i) $A \rightarrow \alpha B \beta$
$\alpha, \beta \in V_T^*$; $A, B, A_1, \ldots, A_p \in V_N$; $p \leq k$

(ii) $A \rightarrow \alpha$

(iii) $S \to A_1 A_2 \ldots A_p$

For any k-linear grammar G we will show how to construct $M \in 2DPDA(3)$ so that $T(M) = L(G)$. Number the type (i) and type (ii) rules of G so that ϵ-productions are first, type (ii) productions next and type (i) productions following the type (ii) productions. Suppose that there are N such rules. $\Gamma_M = \{1,*,T,Z_0,Z_1,\ldots,Z_n\}$ and Z_0 is the bottom of the store.

M will parse inputs x in following manner:

1. Type (ii) productions will be tried sequentially.

2. If the type (iii) production currently being checked is $S \to A_1 A_2 \ldots A_p$, M will search x for the shortest A_1 and record, on the store, sufficient information to return and look for a longer substring if necessary. If an A_1 has been found, M moves its read heads past the A_1 in the input and looks for the shortest A_i, $2 \leq i \leq p$. If for some $i \neq 1$, M fails to find A_1 in the remainder of x, it returns to A_{i-1}. If $i = 1$ when this happens, the next type (iii) rule is tried. If M has found A_p, but the input has not all been examined, then M looks for a longer A_p, backing up to A_{p-j}, $1 \leq j \leq p$, as necessary.

Note that in any derivation from a k-linear grammar, there are at most k unexpanded non-terminals. These non-terminals, as well as which type (iii) production is being tested, can be remembered in the finite control of M. If the store of M contains

$$Z_0 1111T1111111T1111Z_5Z_3$$

then M has found component A_1 of length 4, component A_2 of length 6 and is looking for component A_3 of length 4 and has already applied rules 5 and 3 in the search for A_3. The ones are placed on the store as longer components are searched for; the T's are added when a component is found. The information on the store is used to reposition heads 1 and 2 when M returns to A_{i-1}; head 3 is necessary to avoid destruction of the store during this repositioning.

Proof.

1. M sets I, J, and K equal to 1, moves head 1 one square right and enters step 2 looking for component K of rule J, the search beginning with type (i) or (ii) rule I. Heads 2 and 3 move together during steps 2, 3 and 7 where reference is made to steps of Theorem 12, whose machine has only two heads.

2. M searches for an applicable rule as in step 2, Theorem 12. If M finds an applicable rule it goes to step 7; otherwise, it enters step 3.

3. The store contains $Z_0 1^{i_1} T 1^{i_2} T \ldots T 1^{i_k} Z_{j_1} \ldots Z_{j_p}$. This means M was looking for component K of length i_k and has applied rules $j_1,\ldots,j_p$ in the search. M erases Z_{j_p} and uses this rule to reposition all heads to their positions when rule j_p was selected. This repositioning is accomplished like that in step 4 of Theorem 12. If $j_p < N$, the number of type (i) and (ii) rules, step 2 is entered with $I = j_p + 1$. If $j_p = N$ and $p = 1$, step 4 is entered. Otherwise, step 3 is reentered.

4. The store contains $Z_0 1^{i_1} T 1^{i_2} T \ldots T 1^{i_k}$. M was unable to find component K of length i_k. Head 1 is at the left end of the length i_k

substring, heads 2 and 3 at the right end of this substring. Heads 2 and 3 move one square right and a 1 is added to the store to indicate that M will look for component K of length $i_k + 1$. If heads 2 and 3 do not scan \$, M can reenter step 2 with I = 1, looking now for a longer component K of rule J. Otherwise M enters step 5.

5. In this step, no component K of the proper length can be found. M erases 1's on the store until a Z_0 or T is encountered, moving heads 2 and 3 one square left for each erasure. If a T is encountered, all three heads of M are one square past the right end of a component K - 1 which has previously been found. This component was not of the proper length to allow M to parse the remainder of x using rule J. M replaces the T with a 1 and will now look for a longer component K - 1. Step 6 is entered with K = K - 1. If Z_0 is encountered, then rule J cannot be applied. If J was the last type (iii) rule, M rejects; otherwise, M moves heads 2 and 3 one square left and enters step 2 with J = J + 1, I = 1 and K = 1.

6. When this step is entered, all three heads are scanning the right end of a substring which must be checked to see if it can be generated by G as a component K. The length of this substring is depicted in a unary count of 1's on the store. Using and erasing this count, head 1 is positioned one square to the left of the left end of the substring while heads 2 and 3 remain at the right end. This count must be replaced before step 2 is entered. Head 3 moves left one square and a 1 is added to the store; heads 1 and 3 now move left until head 1 scans a ¢, a * being added to the store for each left move. If head 1

scans ¢ before head 3, the count is not yet correct. The *'s on the store are used to restore heads 1 and 3 to their previous positions. Head 3 moves one square left, M adds a 1 to the store and the left movement toward ¢ described above resumes. When heads 1 and 3 reach ¢ at the same time, the count has been restored. Head 3 must now be moved back to the same square as that scanned by head 2. The *'s are erased, heads 1 and 3 moving right one square for each *. Heads 1 and 3 are now scanning the same square. Head 3 moves one square right and the process described to position heads 1 and 3 on the square scanned by head 1 is modified in the obvious manner to position heads 2 and 3 on the square which is scanned by head 2. The count in 1's is unchanged during this positioning, however. M moves head 1 one square right, sets I = 1 and enters step 2.

7. M checks to see if component K has been found similar to step 3 of Theorem 12. This involves moving all heads left toward ¢, adding a * to the store for each move left. M notes whether (i) the heads "crossed," (ii) the component was not found or (iii) the component was found. The unary count (in *'s) is used to reposition the heads. M now enters step 3 if the heads "crossed," step 2 with I = 1 if the component was not found or step 8 is the component was found.

8. Component K has been found, and M must move all heads to the right end of this component. The sequence of productions used to get component K is on the store, and this record can be used to move heads 1, 2 and 3 to the right end of component A_K. For example, if the top of the store is $\ldots Z_2 Z_5 Z_8$, and rule 8 is $A \rightarrow a_1 a_2 \ldots a_p B b_m \ldots b_1$, then all

three heads are moved m squares right and Z_8 is erased. The remainder of the production rules represented on the top of the store can be used to correctly position the heads to the right end of the component just found. When a 1 is encountered on the store, M moves all three heads one square right and adds a T to the top of the store. The length of component K has been saved in a unary count under T. If the heads are not scanning \$, M replaces the T on the store with a 1 and enters step 6. If the heads scan \$, three cases arise. If K is the last component of a type (iii) rule J, M accepts x. If K is not the last component in rule J, M must check to see if the remaining components in rule J are the right side of an ϵ-production in G. If so, M halts and accepts x; if not, M enters step 5. Q.E.D.

COROLLARY 13.1. If L is a metalinear language, there exists $M \in$ 2DPDA(3) such that T(M) = L.

Proof. Since any metalinear language is k-linear for some $k \geq 1$, this corollary is a trivial consequence of Theorem 13. Q.E.D.

CHAPTER IV

CONCLUDING REMARKS

In this thesis we have examined universal machines and machine hierarchies (Chapter II) and two-way deterministic pushdown acceptors for certain CFL (Chapter III). Several interesting problems for additional study are suggested by the results which have been presented here.

Although we have shown that the class 2DPDA contains a member U universal over 1NFA, it remains open whether 2DPDA is the smallest class containing such a member. One good candidate for a smaller class containing the required universal machine is the class of 2DPDA without endmarkers. Our construction (Theorem 1), however, requires at least one endmarker and cannot be easily modified to provide a 2DPDA without endmarkers which is universal over 1NFA.

For each $h \geq 1$, the results of Theorems 4 and 5 definitively settle the issue of universal machine membership for the classes 2DPDA(h) and 2NPDA(h). However, these results do not provide diagonal language acceptors for these classes. It is of interest to find the smallest integer h' so that the class 2DPDA(h+h') (or 2NPDA(h+h')) contains a machine which can recognize a diagonal language for the class 2DPDA (or 2NPDA(h)).

For each $h \geq 2$, the question of universal machine membership for the classes 2DFA(h) and 2NFA(h) is not definitively settled. In fact,

two different classes, namely 2DPDA(h-1) and 2NPDA(h+1) are shown to contain the required universal machines. It is not known whether the pairs of classes are comparable and we hope to examine this issue. Additionally, the question of whether 2NFA(h) is a proper subclass of 2NFA(h+h') for any h' remains open. The addition of some fixed number of heads to our universal machine of Theorem 9 may allow the construction of a diagonal language acceptor for 2NFA(h).

Another problem suggested by the results of Chapter II is whether similar results are obtainable for various calsses of multihead SA. It is not obvious that Theorems 4 and 5 can be modified to provide an easy answer to this question.

2WPDA is now the smallest class known to contain a machine universal over 2NPDA(h) for all h. We have also directly constructed a language which is 2WPDA recognizable but is not the acceptance set of any multihead PDA. Direct arguments such as these are not in vogue; we have shown that they can, nevertheless, be used to establish relevant, interesting results.

Finally, it remains open whether the classes 2DPDA and 1NPDA are comparable. The results of Chapter III attack this problem. We hope to extend these results to arbitrary CFL in the future, making it possible to construct for any CFL, L, a machine M in 2DPDA(h), $h \leq 13$, so that $T(M) = L$.

BIBLIOGRAPHY

1. Aho, Hopcroft and Ullman (1968), "Time and tape complexity of pushdown automaton languages," *Inf. and Control 13,* 186-206.

2. Aho and Ullman (1969), "The theory of languages," *Math. Systems Theory 2,* 97-125.

3. Bar-Hillel, Perles and Shamir (1961), "On formal properties of simple phrase structure grammar," in *Language and Information,* Y-Bar-Hillel (ed.), Addison-Wesley, Reading, Mass., 116-150.

4. Chomsky (1959), "On certain formal properties of grammars," *Inf. and Control 2,* 137-167.

5. Cook (1971), "Characterization of pushdown machines in terms of time-bounded computers," *JACM 18,* 4-18.

6. Ginsberg, Greibach and Harrison (1967), "Stack automata and compiling," *JACM 14,* 172-201.

7. Ginsberg (1966), *The Mathematical Theory of Context-Free Languages,* McGraw Hill, New York.

8. Gray, Harrison and Ibarra (1967), "Two-way pushdown automata," *Inf. and Control 11,* 30-70.

9. Haines (1964), "A note on the complement of a (minimal) linear language," *Inf. and Control 7,* 307-314.

10. Harrison and Ibarra (1968), "Multitape and multihead pushdown automata," *Inf. and Control 13,* 433-470.

11. Hennie and Stearns (1966), "Two tape simulation of multitape turing machines," *JACM 13,* 533-546.

12. Hopcroft and Ullman (1969), *Formal languages and their relation to automata,* Addison-Wesley, New York.

13. Ibarra (1973), "On two-way multihead automata," *JCSS 7,* 28-33.

14. Kain (1972), *Automata theory: machines and languages,* McGraw-Hill, New York.

15. Kain (1973), Personal Communication.

16. Kameda (1972), "Pushdown automata with counters," *JCSS 6,* 138-150.

17. Knuth and Bigelow (1967), "Programming languages for automata," *JACM 14,* 615-635.

18. Kuroda (1964), "Classes of languages and linear bounded automata," *Inf. and Control 7,* 207-233.

19. Mager (1969), "Writing pushdown acceptors," *JCSS 3,* 276-318.

20. Martin and Gwynn (1973), "Two results concerning the power of two-way deterministic pushdown automata," *Proc. ACM National Conf.,* Atlanta, 342-345.

21. Owings and Feldman (1973), "A class of universal linear bounded automata," *Information Sciences 6,* 187-190.

22. Rabin (1952), "Two-way finite automata," *Proc. Summer Institute of Symbolic Logic at Cornell,* 366-369.

23. Rahimi (1968), "Transformational mappings of computing machine models," Ph.D. thesis, University of Iowa, Iowa City.

24. Rosenberg (1966), "On multihead finite automata," *IBM Journal Res. and Devel. 20,* 388-394.

25. Solamaa (1973), *Formal Languages,* Academic Press, New York.

26. Sheperdson (1959), "Reduction of two-way automata to one-way automata," *IBM Journal Res. and Devel. 3,* 198-200.

VITA

Daniel P. Martin was born in Lynn, Massachusetts on September 27, 1946. After graduating from Carteret High School, Carteret, New Jersey, in 1964, he attended the Georgia Institute of Technology 1964-1968 and 1970-1974. He received the degrees Bachelor of Applied Mathematics (1968), Master of Science in Information and Computer Science (1971), and Doctor of Philosophy in Information and Computer Science (1974) from the Georgia Institute of Technology.

Mr. Martin was employed as an associate manufacturing engineer with the Lockheed-Georgia Company from 1968 to 1970, where he received a Certificate of Personal Excellence and Department Employee-of-the-Month awards. He also has been employed as a computer operator-in-charge at the Rich Electronic Computer Center at the Georgia Institute of Technology and as computer consultant with the Warren-Scherer Company.

Mr. Martin presented a paper before the National Conference of the Association for Computing Machinery in August, 1973.